LOSS-AWARE TEST-TIME SUBSPACE ADAPTATION FOR REDUCED-RANK LLMS

Anonymous authors
Paper under double-blind review

ABSTRACT

Low-rank compression reduces deployment memory but fixes the model subspace before the request is known. We ask whether a deployed reduced-rank language model can adapt its stored factors at inference time without reconstructing or retaining dense weights. We introduce request-local loss-aware test-time adaptation (TTA) in the deployed low-rank subspace: observed prompt tokens define a masked causal loss, request gradients choose update support, and a hard Kullback–Leibler (KL) trust region limits functional drift. Across Qwen3 models, test-time adaptation consistently improves held-out WikiText likelihood and improves GSM8K in selected gauge/support settings, while ARC-Challenge chat remains statistically unchanged despite substantial prompt-loss reduction. Matched-KL ablations further show that gauge, factor side, and block support can change downstream behavior even when functional displacement is nearly fixed. Together, these results identify update geometry and surrogate–task alignment as central design variables for adapting compressed models at test time.

1 INTRODUCTION

Large language models are increasingly deployed under memory constraints that motivate post-training compression. Low-rank factorization is attractive because it replaces a dense linear operator with structured factors and can be combined with activation-aware objectives tailored to observed calibration activations (Yuan et al., 2023; Wang et al., 2024). Yet conventional deployment treats decomposition as a one-time decision: calibration data determine a reduced representation offline, and that representation remains fixed when future requests arrive.

This paper asks a complementary question: *can the deployed low-rank model adapt its own subspace at test time?* The setting is deliberately stricter than parameter-efficient fine-tuning. After offline decomposition, the original dense weights are not needed for inference or adaptation. A test request supplies a masked causal language-model loss, and only the stored reduced-rank factors may change before generation. This creates an opportunity—request-specific information may recover behavior that an offline calibration set did not prioritize—but also a risk: optimizing the visible prompt can overfit a surrogate objective and alter autoregressive behavior in harmful directions.

We develop a framework for **loss-aware test-time subspace adaptation** that begins from an already deployed low-rank model, independent of how its offline factors were obtained. The new question begins *after deployment*: how should the stored low-rank coordinates adapt to the current request? Because $AB = (AG)(G^{-1}B)$ for any invertible G, the same exact low-rank operator admits many factor coordinates; when a block-identity gauge is used, the deployed chart also becomes an optimization coordinate system. We update only the stored factors, with no dense-weight reconstruction, constrain functional drift by a hard KL trust region with deterministic bisection, and score transformer blocks from the request gradient to choose all-block, top-k, or soft support.

Our experiments reveal a recurring distinction between fitting and downstream behavior. At nearly identical realized KL, different block supports can produce materially different GSM8K accuracy, and larger prompt-loss reduction can coincide with worse exact-match accuracy. This distinction persists across tasks. On held-out WikiText, where evaluation is directly aligned with next-token likelihood, both all-block and selective TTA improve NLL. On the canonical zero-shot ARC-Challenge chat protocol, the same frozen TTA configuration substantially improves the observed-token surrogate yet leaves exact-match accuracy statistically unchanged. In a cross-scale GSM8K study, aggressive

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

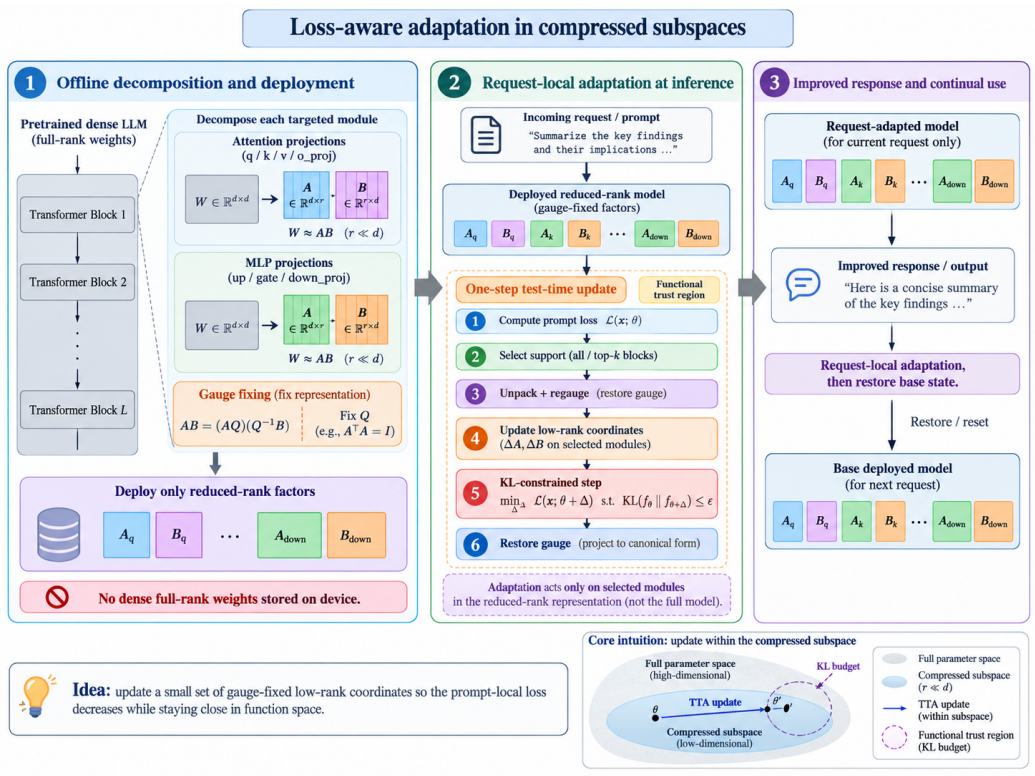

Figure 1: Conceptual overview of loss-aware test-time subspace adaptation. Offline decomposition produces reduced-rank factors; the block-identity gauge used in our principal experiments defines a memory-efficient deployment chart. At test time, the observed request defines a masked causal loss, block scores select adaptation support, and a hard KL trust region limits functional drift before generation; the checkpoint is restored after each request. The gauge and response drawings are schematic, and the exact LU-selected construction is defined in Sec. 3 and Appendix B.3. *This panel is an AI-generated conceptual illustration for explanatory purposes only; it is not empirical evidence or measured system output.*

compression creates a regime in which one-step adaptation does not recover the task performance lost to compression, whereas selected higher-retention settings permit measurable request-specific recovery. Under the same stored-memory budget, the gauge-fixed LatentLLM representation also preserves substantially more task accuracy than native SVD, ASVD, and SVD-LLM baselines.

Our contributions are:

- a request-local TTA formulation that operates entirely on a deployed reduced-rank language model, without retaining the original dense weights;
- a gauge-aware analysis of deployed low-rank factors, building on the block-identity gauge introduced in LatentLLM (Koike-Akino et al., 2026), including intrinsic-coordinate storage, factor-dependent update directions, and memory-efficient gauge restoration;
- a hard KL trust-region procedure with deterministic bisection and a local Fisher analysis that motivates gradient top-k support under a common-curvature approximation; and
- evidence across decompositions, Qwen3 scales, memory budgets, and three evaluation settings that downstream behavior depends on update geometry and surrogate–task alignment, not only on surrogate-loss decrease or KL magnitude.

2 Related Work

Post-training low-rank compression. Classical SVD gives the best rank-constrained approximation in Frobenius norm (Eckart & Young, 1936), but a weight-only objective need not preserve a transformer's activation-weighted function. Recent LLM-specific methods therefore incorporate calibration statistics: ASVD rescales weight matrices using activation statistics and searches layer sensitivities (Yuan et al., 2023), while SVD-LLM uses truncation-aware data whitening and post-truncation low-rank updates (Wang et al., 2024). LatentLLM introduced an activation-aware latent compression framework and the block-identity gauge used here to remove low-rank factor redundancy (Koike-Akino et al., 2025; 2026). The present request-time method is not tied to LatentLLM: it can begin from factors produced by any offline low-rank decomposition, while the gauge-aware experiments use LatentLLM's canonical chart to study how equivalent stored coordinates affect adaptation. These methods complement quantization, pruning, and structured compression approaches such as GPTQ, SparseGPT, SmoothQuant, and SliceGPT (Frantar et al., 2022; Frantar & Alistarh, 2023; Xiao et al., 2023; Ashkboos et al., 2024). Our focus is different: after an offline decomposition has produced the deployed reduced-rank model, we adapt only that deployed representation at inference time.

Low-rank adaptation and adaptive subspaces. LoRA parameterizes task updates in a low-rank form (Hu et al., 2021); AdaLoRA allocates low-rank budget according to parameter importance (Zhang et al., 2023); PiSSA and DoRA alter the initialization or parameterization of low-rank adaptation (Meng et al., 2024; Liu et al., 2024). Low intrinsic dimension in language-model fine-tuning further motivates restricted update spaces (Aghajanyan et al., 2021). In contrast, our model is already low-rank at deployment: the original dense weights are not retained, and test-time optimization moves the compressed factors themselves. We additionally allocate the update across transformer blocks on a per-request basis.

Test-time and online adaptation. Test-time training and fully test-time adaptation update parameters from test inputs using self-supervised or confidence objectives (Sun et al., 2020; Wang et al., 2021; Zhang et al., 2022). Later work addresses forgetting, instability, continual shifts, and memory cost (Niu et al., 2022; Wang et al., 2022; Niu et al., 2023; Song et al., 2023); TTT-NN extends test-time training to language models using retrieved neighboring text (Hardt & Sun, 2024). Our setting instead adapts only compressed low-rank factors, constrains functional drift by KL, and allocates updates across blocks.

Online subspaces and trust regions. Classical methods such as Oja's rule, PAST, GROUSE, and PETRELS update low-dimensional subspaces as samples arrive (Oja, 1982; Yang, 1995; Balzano et al., 2010; Chi et al., 2013). We share the idea of adapting a low-dimensional representation online, but the object here is the deployed factorization of transformer weights and the signal is a request-local language-model loss. Our hard KL constraint is also related in spirit to trust-region optimization (Schulman et al., 2015): it normalizes functional displacement after the update direction and support have been chosen, allowing those choices to be compared at nearly matched predictive displacement. Appendix A gives the detailed positioning.

3 Method: Request-Local Adaptation of Low-Rank Factors

Starting point: any deployed low-rank factorization. We begin from selected linear maps already represented offline as $W \approx AB$, with $A \in \mathbb{R}^{m\times r}$ and $B \in \mathbb{R}^{r\times n}$. The factors may come from SVD, activation-aware decomposition, SVD-LLM-style compression, LatentLLM, or another low-rank method; the request-time formulation below does not require the original dense operator. Our problem begins *after* this compression step: how should the deployed low-rank coordinates be represented and adapted for the current request?

Gauge fixing is a storage and coordinate choice. A rank-r factorization is non-unique:

$$AB = (AG)(G^{-1}B), \qquad G \in \mathrm{GL}(r). \tag{1}$$

Thus storing two unrestricted factor matrices uses $r(m+n)$ scalars even though an exact rank-r operator has only $r(m+n-r)$ intrinsic continuous degrees of freedom. Following the block-identity

gauge introduced in LatentLLM (Koike-Akino et al., 2026), we can fix this r^2-dimensional freedom and store $r(m+n-r)$ floating-point coefficients (plus discrete pivot metadata):

$$N_{\text{gauge}} = r(m+n-r) = r(m+n) - r^2, \tag{2}$$

removing exactly r^2 floating-point coordinates relative to two unrestricted factor matrices. This is purely a change of coordinates: in exact arithmetic it leaves AB unchanged. The equal-memory consequence is immediate. For a factor-storage budget M, an unrestricted pair and a gauge-fixed pair obey

$$r_{\text{pair}}(m+n) \le M, \qquad r_{\text{gauge}}(m+n-r_{\text{gauge}}) \le M. \tag{3}$$

Thus the same budget can support a larger retained rank after gauge fixing (subject to the matrix dimensions); equivalently, at fixed rank the fractional factor-storage saving is $r/(m+n)$. Native SVD/ASVD/SVD-LLM deployments in our comparison retain two unrestricted factor matrices, whereas the block-identity LatentLLM chart can use the saved coordinates to retain a larger rank under the same budget. This storage advantage is only one contributor to the final accuracy gap—offline decomposition quality also matters. Appendix B gives the intrinsic-dimension argument and Appendix B.3 gives the LU pivot construction.

A-side and B-side gauges are different deployment coordinates. Let I be r selected rows of A and J be r selected columns of B, with nonsingular pivot blocks $P_A = A[I,:]$ and $P_B = B[:,J]$. Two useful canonical charts are

$$\text{A-side:} \quad (A_{g_A}, B_{g_A}) = (AP_A^{-1},\ P_A B), \qquad A_{g_A}[I,:] = I_r, \tag{4}$$

$$\text{B-side:} \quad (A_{g_B}, B_{g_B}) = (AP_B,\ P_B^{-1} B), \qquad B_{g_B}[:,J] = I_r. \tag{5}$$

The naming refers only to *where the identity block lives*; it does *not* restrict which factor may be optimized. Under either gauge we may update A, B, or both. The implementation difference is whether a single-factor step preserves the canonical identity block automatically: under A-side gauge, a B-only step is already gauge preserving, whereas an A-only step is followed by an exact regauging; under B-side gauge, the roles reverse.

Gauge	fixed identity block	fully stored factor	no-regauge single-factor step
A-side	$A[I,:] = I_r$	B	update B
B-side	$B[:,J] = I_r$	A	update A

Coupled projections create a gauge/update design space. The gauge can be chosen independently for coupled projection groups. For a SwiGLU MLP we treat gate/up as one group and `down_proj` as another, so $(g_{\text{gu}}, g_{\text{down}}) \in \{A,B\}^2$ gives the four AA/AB/BA/BB deployments evaluated in Sec. 4. The same construction applies to other paired groups such as value/output projections. With G independently gauged groups, the combined design space is

$$g \in \{A,B\}^G, \qquad u \in \{A, B, A{+}B\}, \qquad \mathcal{B}(x) \subseteq \{1,\dots,L\}, \tag{6}$$

where g is the offline gauge assignment, u is the factor coordinate updated in our experiments, and $\mathcal{B}(x)$ is the request-dependent set of transformer blocks. For the two MLP groups, the 4×3 gauge/update choices are exactly the 12 cells in our matched mechanism study. This is the central coordinate-design problem studied here.

The factor side is not merely an implementation detail. If $H = \partial \mathcal{L}/\partial W$, then, ignoring scalar balancing for clarity, one Euclidean factor-gradient step induces

$$\Delta W_A = -\eta\, H(B^\top B), \qquad \Delta W_B = -\eta\, (AA^\top)H. \tag{7}$$

Thus factor side and non-orthogonal gauge change the induced operator direction. Appendix B derives the joint-factor and tangent-space views; Sec. 4 tests these choices at matched KL.

What this paper adds at request time. For an incoming request x, let $\mathcal{S}_{g,u,\mathcal{B}}(x)$ denote the update coordinates determined by the deployed gauge assignment g, factor-update choice u, and selected transformer blocks $\mathcal{B}(x)$. We adapt only those deployed coordinates using observed request tokens:

$$\min_{\Delta \in \mathcal{S}_{g,u,\mathcal{B}}(x)} \mathcal{L}_{\text{req}}(\theta + \Delta; x) \quad \text{s.t.} \quad D_{\text{KL}}(p_\theta \,\|\, p_{\theta+\Delta}) \le \varepsilon. \tag{8}$$

The request loss uses only observed causal targets. With $\mathcal{M}(x)$ denoting their positions,

$$\mathcal{L}_{\text{req}}(\theta; x) = -\frac{1}{|\mathcal{M}(x)|} \sum_{t \in \mathcal{M}(x)} \log p_\theta(x_t \mid x_{<t}). \tag{9}$$

For GSM8K, $\mathcal{M}(x)$ contains demonstration-answer tokens and observed query tokens, never the unknown answer; other task masks obey the same no-label-leakage principle (Appendix B). The hard KL constraint then separates gauge, factor-side, and block-support effects from step-size effects.

Why gradient top-k? The KL constraint normalizes how far a chosen update moves the model, but it does not choose which blocks should move. Locally, with request gradient g and predictive Fisher F, the best first-order loss decrease available on a block support S scales as

$$\Delta\mathcal{L}_{\max}(S) \approx \sqrt{2\varepsilon\, g_S^\top F_S^{-1} g_S}; \qquad F_b \approx cI \implies S^* = \text{TopK}_b \|g_b\|_2. \tag{10}$$

Thus gradient-norm top-k is the inexpensive support rule obtained under a common-curvature block approximation, while the actual algorithm still checks the true nonlinear KL by deterministic bisection (Amari, 1998; Schulman et al., 2015). Appendix B gives the full derivation and its assumptions.

Per-request algorithm. The deployed procedure is intentionally simple:

1. compute the masked request loss and gradients of the stored low-rank factors;
2. score transformer blocks from their request-gradient norms and choose all blocks or a top-k subset;
3. take one balanced low-rank factor step in the chosen gauge/update coordinates, regauging only if the identity-bearing factor moved;
4. deterministically bisect one global scale until the true nonlinear KL lies just inside ε; and
5. generate with the adapted model, then restore the fixed compressed checkpoint before the next request.

Our principal B/B + factor-A deployment stores A fully in both MLP groups, so an A-only step preserves the canonical gauge without regauging. Section 4 evaluates the resulting accuracy–cost tradeoff. Appendices B and B.3 give the Fisher motivation, balancing, finite-precision analysis, operator projection, and pseudocode.

4 Experiments and Results

We use deterministic GSM8K exact match (Cobbe et al., 2021) for decomposition and mechanism studies, WikiText likelihood for language-model transfer and the main compression-scale sweep, and ARC-Challenge for a second reasoning task. All experiments use Qwen3 (Qwen Team, 2025). Unless stated otherwise, the principal TTA configuration uses one request-local update, learning rate 0.0015, and a hard KL budget of 0.025; selective runs use the top 50% of blocks by request-gradient norm. We control deployment by stored-parameter fraction rather than matrix rank because memory is the intended resource constraint.

4.1 Experimental setup

Models, calibration, and deployment. Our principal studies use Qwen3-0.6B, 1.7B, and 4B in bfloat16 (BF16). The decomposition and GSM8K mechanism studies calibrate selected MLP projections from 4,096 WikiText tokens (Merity et al., 2016); the separate likelihood-scale study intentionally calibrates from the Colossal Clean Crawled Corpus (C4) (Raffel et al., 2020) and evaluates on WikiText to test corpus transfer. For compressed runs, evaluation and TTA use only the stored reduced-rank representation. The principal gauge is B/B with factor-A adaptation, chosen before the final cross-task transfer experiments. Unless a study explicitly isolates gauge or factor coordinates, the same TTA hyperparameters are transferred without per-model or per-task retuning.

Table 1: Decomposition-method comparison at 0.95 stored-parameter fraction. Values are GSM8K exact-match accuracy (%).

Model	Method	Fixed	All	Top-k
0.6B	SVD	0.00	0.00	0.00
	ASVD	0.83	1.21	1.21
	SVD-LLM	1.67	3.03	2.88
	LatentLLM	32.07	34.27	**34.65**
1.7B	SVD	0.00	0.00	0.00
	ASVD	19.86	21.46	21.83
	SVD-LLM	14.71	15.54	15.09
	LatentLLM	61.94	63.53	**64.59**

Tasks and metrics. GSM8K uses deterministic exact match on the full 1,319-example test set. WikiText transfer evaluates held-out next-token negative log-likelihood (NLL) and perplexity under the same frozen request-local update. For a second reasoning task, we use the canonical zero-shot `arc_challenge_chat` generation protocol for ARC-Challenge (Clark et al., 2018), following the Language Model Evaluation Harness (Biderman et al., 2024): the model receives the question and choices but not the answer, generates the answer continuation, and is scored by exact match after the harness-compatible filter. Our dense Qwen3-1.7B parity run reaches 73.12%, within 0.59 percentage points of an independent harness run (72.53% $\pm$ 1.30 points), before evaluating compressed/TTA variants.

Determinism and statistics. Deterministic runs use fixed seeds, deterministic CUDA settings, and no stochastic decoding on NVIDIA A40 GPUs. We use paired exact McNemar tests for discrete predictions on the same examples. Adaptation and generation times are reported separately; TTA time is an implementation measurement, not optimized serving throughput. Appendix C gives masks, determinism controls, and hardware metadata.

4.2 Equal-memory decomposition comparison

At 0.95 stored-parameter fraction, we compare native SVD, ASVD, SVD-LLM, and prior LatentLLM compression (Koike-Akino et al., 2026) under one frozen TTA protocol.

For LatentLLM, top-k gains $+2.58$ points at 0.6B and $+2.65$ at 1.7B; paired exact McNemar tests give $p = 0.029$ and $p = 0.0076$. ASVD also improves selectively at 1.7B, so request-adaptive support is not unique to the gauge-fixed representation. Yet the much larger fixed-model gap shows that offline representation remains a dominant choice under equal memory. Appendix C reports paired comparisons and per-method KL/runtime measurements.

4.3 Gauge and factor-update coordinates

The factor-coordinate analysis in Appendix B predicts that algebraically equivalent low-rank gauges need not define equivalent Euclidean TTA steps. We test this prediction in a matched deterministic Qwen3-0.6B experiment using the principal 4,096-token WikiText calibration, stored-parameter fraction 0.95, all-block adaptation, and hard KL budget 0.025. Table 2 crosses the gate/up–down gauge (AA/AB/BA/BB) with factor-A, factor-B, and joint A+B updates.

Table 2: Matched gauge $\times$ update-coordinate GSM8K accuracy (%). Parentheses give change from the corresponding fixed model. Adaptive cells have realized KL in the narrow range 0.02035–0.02284.

Gauge	Fixed	A update	B update	A+B update
AA	32.22	32.75 ($+0.53$)	30.40 (-1.82)	29.87 (-2.35)
AB	32.68	33.66 ($+0.99$)	32.37 (-0.30)	33.97 ($+1.29$)
BA	31.84	31.61 (-0.23)	29.64 (-2.20)	29.04 (-2.81)
BB	32.07	34.27 ($+2.20$)	33.97 ($+1.90$)	**34.72** ($+2.65$)

The matched KL makes the coordinate effect explicit: accuracy changes span -2.81 to $+2.65$ points despite similar scalar functional displacement. Even before adaptation, the fixed BF16 realizations differ modestly (31.84–32.68%), consistent with the finite-precision analysis in Appendix B; exact algebraic equivalence does not imply identical rounded factors. BB is the only gauge for which all three factor-update modes improve over the corresponding fixed model. Joint BB A+B reaches the highest all-block accuracy (34.72%), but BB+A is the practical factor-coordinate choice: it reaches 34.27% in 1.332 s/problem, whereas BB+A+B requires 6.428 s/problem for an additional 0.45 points. We do not attach significance claims to within-matrix differences without paired prediction analysis.

The runtime reversal matches the gauge construction: under AA, B-only avoids regauging (1.208 versus 6.481 s/problem for A); under BB, A-only avoids regauging (1.332 versus 6.266 for B). Prompt fit still fails to rank updates: BA+A improves masked CE by 0.1919 yet loses 0.23 accuracy points, while BB+A+B improves CE less than BB+A (0.1537 versus 0.2348) but scores slightly higher. Appendix C gives the full ΔCE, KL, runtime, acceptance, and earlier WikiText diagnostics. These results motivate B/B + factor-A: it is empirically effective and is the no-regauge single-factor path under B-side gauge.

Gauge-invariant reference control. Appendix B also constructs an operator-space reference that implicitly forms the unconstrained operator-gradient step and then computes its exact best rank-r projection through skinny QR factors and a small-core SVD, without materializing the full trial matrix. This removes factor-gauge dependence and asks whether the practical coordinate restriction is hiding an obviously better operator-space step. In a 128-example diagnostic, operator projection showed no accuracy advantage over the practical B/B + factor-A control while requiring about $91\times$ more TTA time. Because that diagnostic used an auxiliary C4-calibrated representation, we treat it only as a mechanism/runtime control rather than a main accuracy comparison.

4.4 Update support matters beyond KL magnitude and prompt fit

The strongest support-selection evidence comes from an independent frozen 1.7B scale sweep at stored-parameter fraction 0.95. The small numerical differences from Table 1 reflect the independent frozen sweep rather than retuning. Table 3 compares all-block and hard top-k updates. Their realized KL is effectively identical and all-block adaptation decreases prompt CE slightly more, yet top-k is 1.29 points higher in exact match while adapting half as many blocks and reducing measured TTA time by about 27%.

Table 3: Support comparison in the frozen Qwen3-1.7B, 0.95 stored-parameter-fraction sweep. ΔCE is masked prompt cross-entropy improvement.

Policy	Acc. (%)	ΔCE	Realized KL	Blocks	TTA s/problem
Fixed	61.94	–	–	–	0.00
All blocks	63.38	**0.1503**	0.02178	28	3.72
Top-k	**64.67**	0.1488	0.02178	14	**2.73**

An earlier 0.6B B/B + factor-A diagnostic reinforces this point: increasing KL from 0.02 to 0.03 raises ΔCE from 0.1146 to 0.1789 but lowers GSM8K accuracy from 36.39% to 34.27%. Neither surrogate gain nor scalar functional distance therefore identifies a useful direction by itself (Appendix C).

4.5 Cross-task transfer exposes surrogate–task alignment

To test transfer beyond GSM8K, we freeze B/B + factor-A and reuse it without task-specific tuning. Table 4 spans held-out WikiText, GSM8K, and canonical ARC-Challenge chat, covering next-token likelihood, mathematical exact match, and zero-shot science-question generation.

On WikiText, all-block adaptation lowers NLL by 0.0552 at 0.6B and 0.1096 at 1.7B. Top-k retains about 91–92% of those NLL gains while reducing measured TTA time from 1.335 to 1.096 s/problem and from 2.766 to 1.981 s/problem, respectively. This is the expected behavior when the request surrogate and evaluation metric are closely aligned. GSM8K differs: top-k is more accurate than

Table 4: Frozen cross-task transfer. WikiText reports NLL (lower is better); GSM8K and ARC-chat report exact-match accuracy in percent (higher is better). No task-specific TTA retuning is used.

Task	Model / metric	Fixed	All	Top-k
WikiText	0.6B NLL ↓	3.1747	**3.1195**	3.1243
WikiText	1.7B NLL ↓	2.9477	**2.8381**	2.8469
GSM8K	1.7B EM ↑	61.94	63.38	**64.67**
ARC-chat	1.7B EM ↑	**66.21**	65.78	66.13

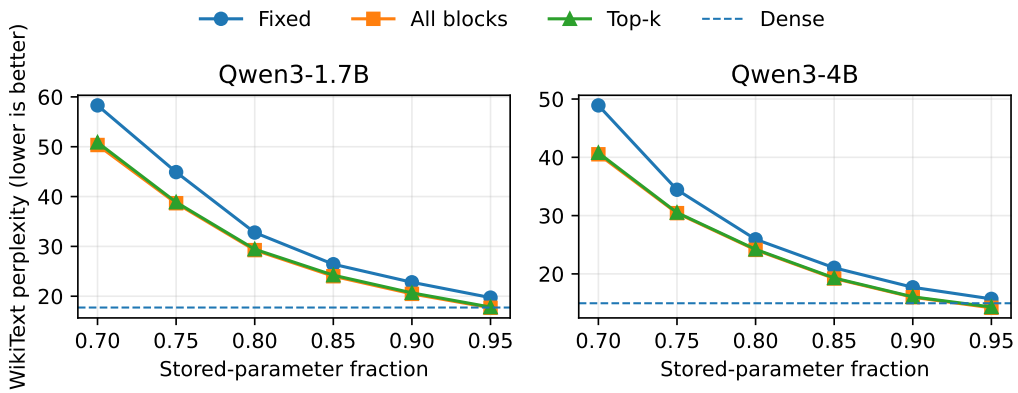

Figure 2: C4-calibrated → WikiText likelihood recovery across stored-parameter fraction. Lower perplexity is better. TTA improves held-out WikiText perplexity at every measured fraction for both model scales; all-block adaptation is best, while top-k uses half the transformer blocks and closely tracks it. The dashed line is the static undecomposed dense model evaluated by the identical continuation scorer.

all-block at essentially the same realized KL, showing that support matters even when both policies improve the surrogate.

ARC-chat provides the complementary boundary case for surrogate–task alignment. Compression lowers the dense 1.7B model from 73.12% to 66.21%. All-block and top-k TTA then improve masked prompt CE by 0.3330 and 0.3412 at matched KL near 0.016, yet exact match remains statistically unchanged. Relative to fixed, all-block produces 17 wrong→right and 22 right→wrong flips (exact McNemar $p = 0.522$), while top-k produces 19 and 20 ($p = 1.0$). Thus a large request-local likelihood improvement can coexist with no downstream exact-match gain. The detailed transfer table, parity check, and paired transitions are in Appendix C.

4.6 Likelihood recovery across compression levels

To isolate the regime in which the request-local objective is aligned with evaluation, we run a separate scale sweep in which the low-rank factors are calibrated from 4,096 C4 tokens and then evaluated on held-out WikiText continuations. Figure 2 uses the same frozen B/B + factor-A update, hard KL budget, and six stored-parameter fractions for Qwen3-1.7B and 4B. Thus the adaptation prompt comes from the evaluation request, while the offline subspace was obtained from a different corpus.

The result is qualitatively different from the GSM8K boundary sweep. Across all 12 compressed model/fraction points, all-block TTA reduces perplexity (PPL) by 7.0–17.1% relative to the fixed compressed model, and top-k reduces it by 6.6–16.6%. Top-k remains within 0.10–0.52 PPL of all-block while updating half as many blocks and reducing measured adaptation time by roughly 24–35%. At stored-parameter fraction 0.95, all-block reaches PPL 17.677 versus 19.721 fixed for Qwen3-1.7B and 14.183 versus 15.711 for Qwen3-4B. These adapted values fall below the corresponding static dense references (17.730 and 14.963). Because the compressed model receives request-specific adaptation while the dense reference remains static, this comparison measures request-

time compensation rather than global checkpoint superiority. Full NLL, KL, ΔCE, block-count, and runtime values are reported in Appendix C.5.

4.7 Compression boundary on GSM8K

The full six-point stored-parameter-fraction sweep is reported in Appendix C (Fig. 4). Its main role is diagnostic: fixed task accuracy falls sharply under severe compression, and one request-local step changes exact match only modestly relative to that representation loss. At the high-retention end, however, Qwen3-1.7B top-k improves the 0.95 fixed baseline by $+2.73$ points (exact McNemar $p = 0.0064$), while the same frozen update is not uniformly beneficial across lower fractions or the 4B model. The Appendix therefore uses this sweep to delimit where reasoning recovery is and is not available, rather than as the main visual evidence for TTA benefit.

Two distinct limits of request-local recovery. Together, the scale and cross-task experiments separate two failure modes that prompt loss alone would conflate. When the surrogate and metric are aligned, the C4$\rightarrow$WikiText sweep shows consistent likelihood recovery across compression levels. Under severe GSM8K compression, by contrast, fixed task accuracy falls sharply and surrogate-improving local steps change accuracy only modestly; ARC-chat retains substantial task ability and achieves a large ΔCE, yet beneficial and harmful answer flips nearly cancel. Recovery therefore requires both an accessible direction in the deployed subspace and alignment between the request-local surrogate and the downstream decision.

Scope and limitations. We deliberately isolate one-step request-local adaptation: every example starts from the same compressed checkpoint, which is restored after generation, so we do not study accumulation across requests. The mechanism studies focus on MLP projection groups; the same gauge construction can be applied to paired attention projections, but a full attention-group search and multi-step adaptation are outside the present scope. Reported TTA times are implementation measurements, not optimized serving throughput; the local Fisher/top-k argument motivates support selection but does not guarantee nonlinear exact-match gains.

4.8 Design implications

The evidence suggests three deployment axes. First, the offline representation determines what survives and how much rank fits in memory; gauge fixing can reclaim redundant coordinates but cannot compensate for a poor decomposition. Second, gauge assignment, factor side, and block support determine the *direction* of the request update, while the hard KL constraint primarily controls its magnitude. Third, the request loss must align with downstream behavior: WikiText benefits when the objective matches evaluation, selective support helps GSM8K, and ARC-chat remains task-neutral despite surrogate improvement. Compression, adaptation geometry, and request objective should therefore be treated as separate design axes rather than collapsed into a single step-size choice.

5 Conclusion

We studied request-local adaptation after the original dense weights have been replaced by reduced-rank deployment factors. A hard KL constraint controls functional drift but not direction: nearly matched-KL updates can yield different downstream accuracy, and prompt loss can improve without better exact match. WikiText improves when the surrogate matches evaluation, GSM8K benefits from selective support, and canonical ARC-chat remains statistically neutral. When block-identity gauge fixing is used, it saves storage while also defining consequential optimization coordinates. Effective recovery thus depends on retained task structure, update geometry, and objective alignment.

AI USE STATEMENT

OpenAI ChatGPT assisted research brainstorming, prose revision, literature organization, code development and debugging, and generation of the non-empirical conceptual illustration in Fig. 1. All experiments were conducted by the authors, primarily on NVIDIA A40 GPU. The authors reviewed all AI-assisted material; no generative AI tool was used to fabricate or alter empirical measurements. The authors take responsibility for the final manuscript.

REFERENCES

Armen Aghajanyan, Sonal Gupta, and Luke Zettlemoyer. Intrinsic dimensionality explains the effectiveness of language model fine-tuning. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing*, pp. 7319–7328, 2021. doi: 10.18653/v1/2021.acl-long.568.

Shun-ichi Amari. Natural gradient works efficiently in learning. *Neural Computation*, 10(2):251–276, 1998. doi: 10.1162/089976698300017746.

Saleh Ashkboos, Maximilian L. Croci, Marcelo Gennari do Nascimento, Torsten Hoefler, and James Hensman. SliceGPT: Compress large language models by deleting rows and columns. In *International Conference on Learning Representations*, 2024.

Laura Balzano, Robert Nowak, and Benjamin Recht. Online identification and tracking of subspaces from highly incomplete information. In *2010 48th Annual Allerton Conference on Communication, Control, and Computing*, pp. 704–711, 2010. doi: 10.1109/ALLERTON.2010.5706966.

Stella Biderman, Hailey Schoelkopf, Lintang Sutawika, Leo Gao, Jonathan Tow, Baber Abbasi, Alham Fikri Aji, Pawan Sasanka Ammanamanchi, Sidney Black, Jordan Clive, Anthony DiPofi, et al. Lessons from the trenches on reproducible evaluation of language models. *arXiv preprint arXiv:2405.14782*, 2024.

Yuejie Chi, Yonina C. Eldar, and Robert Calderbank. PETRELS: Parallel subspace estimation and tracking by recursive least squares from partial observations. *IEEE Transactions on Signal Processing*, 61(23):5947–5959, 2013. doi: 10.1109/TSP.2013.2282910.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Emily L. Denton, Wojciech Zaremba, Joan Bruna, Yann LeCun, and Rob Fergus. Exploiting linear structure within convolutional networks for efficient evaluation. In *Advances in Neural Information Processing Systems*, volume 27, 2014.

Carl Eckart and Gale Young. The approximation of one matrix by another of lower rank. *Psychometrika*, 1(3):211–218, 1936. doi: 10.1007/BF02288367.

Elias Frantar and Dan Alistarh. SparseGPT: Massive language models can be accurately pruned in one-shot. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 10323–10337, 2023.

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. GPTQ: Accurate post-training quantization for generative pre-trained transformers. *arXiv preprint arXiv:2210.17323*, 2022.

Gene H. Golub and Charles F. Van Loan. *Matrix Computations*. Johns Hopkins University Press, Baltimore, MD, 4 edition, 2013.

Moritz Hardt and Yu Sun. Test-time training on nearest neighbors for large language models. In *International Conference on Learning Representations*, 2024.

Nicholas J. Higham. *Accuracy and Stability of Numerical Algorithms*. Society for Industrial and Applied Mathematics, Philadelphia, PA, 2 edition, 2002. doi: 10.1137/1.9780898718027.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

Toshiaki Koike-Akino, Xiangyu Chen, Jing Liu, Ye Wang, Matthew Brand, et al. LatentLLM: Attention-aware joint tensor compression. *arXiv preprint arXiv:2505.18413*, 2025.

Toshiaki Koike-Akino, Xiangyu Chen, Jing Liu, Ye Wang, Pu Perry Wang, and Matthew Brand. LatentLLM: Activation-aware transform to multi-head latent attention. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 22644–22652, 2026.

Shih-Yang Liu, Chien-Yi Wang, Hongxu Yin, Pavlo Molchanov, Yu-Chiang Frank Wang, Kwang-Ting Cheng, and Min-Hung Chen. DoRA: Weight-decomposed low-rank adaptation. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 32100–32121, 2024.

Fanxu Meng, Zhaohui Wang, and Muhan Zhang. PiSSA: Principal singular values and singular vectors adaptation of large language models. *arXiv preprint arXiv:2404.02948*, 2024.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*, 2016.

Bamdev Mishra, Gilles Meyer, Silvère Bonnabel, and Rodolphe Sepulchre. Fixed-rank matrix factorizations and riemannian low-rank optimization. *Computational Statistics*, 29(3–4):591–621, 2014. doi: 10.1007/s00180-013-0464-z.

Shuaicheng Niu, Jiaxiang Wu, Yifan Zhang, Yaofo Chen, Shijian Zheng, Peilin Zhao, and Mingkui Tan. Efficient test-time model adaptation without forgetting. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 16888–16905, 2022.

Shuaicheng Niu, Jiaxiang Wu, Yifan Zhang, Zhiquan Wen, Yaofo Chen, Peilin Zhao, and Mingkui Tan. Towards stable test-time adaptation in dynamic wild world. In *International Conference on Learning Representations*, 2023.

Erkki Oja. Simplified neuron model as a principal component analyzer. *Journal of Mathematical Biology*, 15:267–273, 1982. doi: 10.1007/BF00275687.

Qwen Team. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67, 2020.

John Schulman, Sergey Levine, Pieter Abbeel, Michael I. Jordan, and Philipp Moritz. Trust region policy optimization. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pp. 1889–1897, 2015.

Junha Song, Jungsoo Lee, In So Kweon, and Sungha Choi. EcoTTA: Memory-efficient continual test-time adaptation via self-distilled regularization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11920–11929, 2023.

Yu Sun, Xiaolong Wang, Zhuang Liu, John Miller, Alexei A. Efros, and Moritz Hardt. Test-time training with self-supervision for generalization under distribution shifts. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 9229–9248, 2020.

Bart Vandereycken. Low-rank matrix completion by riemannian optimization. *SIAM Journal on Optimization*, 23(2):1214–1236, 2013. doi: 10.1137/110845768.

Dequan Wang, Evan Shelhamer, Shaoteng Liu, Bruno Olshausen, and Trevor Darrell. Tent: Fully test-time adaptation by entropy minimization. In *International Conference on Learning Representations*, 2021.

Qin Wang, Olga Fink, Luc Van Gool, and Dengxin Dai. Continual test-time domain adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7201–7211, 2022. doi: 10.1109/CVPR52688.2022.00706.

Xin Wang, Yu Zheng, Zhongwei Wan, and Mi Zhang. SVD-LLM: Truncation-aware singular value decomposition for large language model compression. *arXiv preprint arXiv:2403.07378*, 2024.

Guangxuan Xiao, Ji Lin, Mickael Seznec, Hao Wu, Julien Demouth, and Song Han. SmoothQuant: Accurate and efficient post-training quantization for large language models. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 38087–38099, 2023.

Bin Yang. Projection approximation subspace tracking. *IEEE Transactions on Signal Processing*, 43 (1):95–107, 1995. doi: 10.1109/78.365290.

Zhihang Yuan, Yuzhang Shang, Yue Song, Dawei Yang, Qiang Wu, Yan Yan, and Guangyu Sun. ASVD: Activation-aware singular value decomposition for compressing large language models. *arXiv preprint arXiv:2312.05821*, 2023.

Marvin Zhang, Sergey Levine, and Chelsea Finn. MEMO: Test time robustness via adaptation and augmentation. In *Advances in Neural Information Processing Systems*, volume 35, 2022.

Qingru Zhang, Minshuo Chen, Alexander Bukharin, Nikos Karampatziakis, Pengcheng He, Yu Cheng, Weizhu Chen, and Tuo Zhao. AdaLoRA: Adaptive budget allocation for parameter-efficient fine-tuning. *arXiv preprint arXiv:2303.10512*, 2023.

A Detailed Related Work

A.1 Low-rank approximation and LLM compression

The Eckart–Young result characterizes truncated SVD as the optimal rank-r approximation under unweighted Frobenius error (Eckart & Young, 1936). Low-rank factorization has a long history as a neural-network compression tool (Denton et al., 2014); however, modern LLMs make the distinction between weight reconstruction and functional reconstruction especially important because activation distributions are strongly anisotropic and layer sensitivities differ.

ASVD explicitly incorporates activation statistics into the decomposition by transforming weights according to observed activation scales and by searching layer-specific truncation sensitivities (Yuan et al., 2023). SVD-LLM instead derives a truncation-aware whitening transform so that singular-value truncation is more directly related to activation-weighted compression error, followed by sequential low-rank updates to compensate for truncation (Wang et al., 2024). LatentLLM is a prior compression framework: the earlier preprint formulated attention-aware joint tensor compression (Koike-Akino et al., 2025), and the AAAI publication developed the activation-aware transform to multi-head latent attention (Koike-Akino et al., 2026). The present request-time method is decomposition-agnostic and can start from any deployed low-rank factors. Our principal gauge-aware experiments use LatentLLM's block-identity deployment chart, while TTA itself is a contribution of the present work, not of the cited LatentLLM paper. We compare against vanilla SVD, ASVD, and SVD-LLM under a common stored-parameter budget. We do not add a separate "SVD-LLM + Gauge" row: the baseline set compares each method in its native deployment representation, and LatentLLM is the gauge-fixed method in that set.

Low-rank decomposition is one axis of a broader LLM-compression landscape. GPTQ uses approximate second-order information for post-training weight quantization (Frantar et al., 2022); SparseGPT performs one-shot pruning with approximate inverse-Hessian information (Frantar & Alistarh, 2023); SmoothQuant transfers activation outlier difficulty into weights through an equivalent rescaling to enable W8A8 quantization (Xiao et al., 2023); and SliceGPT exploits transformer invariances to delete rows and columns while preserving dense matrix structure (Ashkboos et al., 2024). These

approaches motivate evaluating deployment cost directly rather than treating matrix rank as the resource of interest. Accordingly, our primary comparisons control the stored reduced-rank parameter budget.

A.2 Low-rank adaptation and adaptive allocation

LoRA freezes a dense pretrained model and learns a low-rank update $\Delta W = BA$ (Hu et al., 2021). This differs from our deployment setting, where the low-rank factors constitute the model itself after offline decomposition and the dense matrix is absent. Nevertheless, the PEFT literature provides useful evidence that task adaptation often lives in a low-dimensional space. Aghajanyan et al. (2021) empirically show that language-model fine-tuning has low intrinsic dimension.

Several LoRA variants are particularly related to our allocation mechanism. AdaLoRA represents updates with an SVD-like parameterization and dynamically allocates parameter budget according to importance (Zhang et al., 2023). PiSSA initializes the trainable low-rank component using principal singular vectors of the pretrained weight (Meng et al., 2024). DoRA separates magnitude and direction before applying low-rank adaptation to the directional component (Liu et al., 2024). These methods adapt a source model during downstream training, whereas we compute a request-conditioned loss at inference, score transformer blocks from that request, and spend a fixed update budget only on the selected subspaces.

A.3 Test-time training and adaptation

Test-time training updates a predictor on each test sample using a self-supervised auxiliary task (Sun et al., 2020). Tent established a widely used fully test-time adaptation setting based on entropy minimization and normalization-parameter updates (Wang et al., 2021). MEMO adapts a model on a single test input using augmentation-consistent marginal entropy (Zhang et al., 2022). EATA filters unreliable or redundant test samples and regularizes important parameters to reduce forgetting (Niu et al., 2022); SAR addresses instability under small batches, mixed shifts, and label imbalance (Niu et al., 2023). CoTTA extends adaptation to nonstationary test streams using averaged predictions and stochastic restoration (Wang et al., 2022), while EcoTTA reduces continual-adaptation memory by introducing lightweight adaptation networks and self-distilled regularization (Song et al., 2023).

Most of this literature studies distribution shift in vision and begins with a full source model. TTT-NN is a closer language-model precedent: it adapts an LM at test time using retrieved nearest-neighbor text (Hardt & Sun, 2024). Our question is orthogonal: can a model that has already discarded its dense weights adapt its *compressed representation* using only the current request? The resulting constraints are different. The update must remain within the representational capacity of the deployed factors, must not require reconstruction of the original dense weights, and must be controlled tightly enough to avoid surrogate-loss overfitting during autoregressive reasoning.

A.4 Online subspace tracking and trust-region control

Classical online subspace methods update low-dimensional representations as new samples arrive. Oja's rule provides an online principal-component update (Oja, 1982); PAST tracks a projection subspace using recursive least squares (Yang, 1995); GROUSE performs Grassmannian online subspace estimation from incomplete observations (Balzano et al., 2010); and PETRELS tracks subspaces through parallel recursive least squares (Chi et al., 2013). Our method shares the idea of adapting a low-dimensional representation online, but the signal is a language-model test-time loss and the object being updated is the deployed factorization of transformer weights rather than a data covariance subspace.

We also use a trust-region principle to separate direction selection from functional step size. Trust-region methods constrain policy changes using KL divergence in reinforcement learning (Schulman et al., 2015). Here the KL constraint is not used for policy optimization; it limits the change in the language model's predictive distribution after the proposed low-rank update. Deterministic bisection chooses the largest admissible scale toward a fixed fraction of the KL budget. This turns the KL budget into a directly comparable functional constraint across block-selection policies and factor parameterizations.

A.5 Positioning of the present work

Taken together, these strands leave a setting not directly covered by any one prior line of work. We study a *post-deployment* low-rank model; update its existing factors rather than adding a dense-model adapter; use the observed request to define the loss and adaptation support; analyze block-identity gauge coordinates when they are used for deployment; and impose a hard functional KL constraint to prevent uncontrolled drift. Empirically, the distinction between surrogate fitting and downstream behavior is central: larger prompt-loss improvement does not necessarily imply better GSM8K accuracy, and selective support can outperform all-block adaptation at nearly the same realized KL.

B Method Details and Derivations

B.1 Activation-aware reduced-rank objective

For completeness, this subsection records the activation-aware offline objective used for the LatentLLM representation in our principal gauge-fixed experiments. The request-time method itself assumes only deployed factors A, B and is otherwise decomposition-agnostic. Consider a linear map $W \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$ and a calibration activation matrix $X \in \mathbb{R}^{d_{\text{in}} \times n}$. We seek factors $A \in \mathbb{R}^{d_{\text{out}} \times r}$ and $B \in \mathbb{R}^{r \times d_{\text{in}}}$ that preserve both observed activations and the original operator:

$$\min_{A,B} \ \|WX - ABX\|_F^2 + \alpha \|W - AB\|_F^2, \qquad \alpha > 0. \tag{11}$$

Let

$$C = XX^\top + \alpha I. \tag{12}$$

Because $\alpha > 0$, C is positive definite, and Eq. (11) is exactly $\|(W - AB)C^{1/2}\|_F^2$. Thus the problem is equivalent to approximating $WC^{1/2}$ in Frobenius norm. If

$$WC^{1/2} = U\Sigma V^\top, \tag{13}$$

then a rank-r solution can be written as

$$A_0 = U_r \Sigma_r^{1/2}, \qquad B_0 = \Sigma_r^{1/2} V_r^\top C^{-1/2}, \tag{14}$$

so that $A_0 B_0$ is the activation-aware rank-r operator. In deployment we retain only reduced-rank factors; the dense W is not required by test-time adaptation.

B.2 Gauge freedom, intrinsic dimension, and canonical coordinates

Any exact rank-r factorization with full-column-rank A and full-row-rank B has the invertible gauge freedom

$$AB = (AG)(G^{-1}B), \qquad G \in \mathrm{GL}(r). \tag{15}$$

This is the standard quotient structure of fixed-rank matrix factorizations (Mishra et al., 2014; Vandereycken, 2013). It leaves the exact operator unchanged but changes the numerical coordinates used for storage and Euclidean optimization.

For a B-side block-identity gauge, let J index r pivot columns of B such that $B_{:,J}$ is nonsingular. Choosing

$$G = B_{:,J} \tag{16}$$

produces

$$A_g = AB_{:,J}, \qquad B_g = B_{:,J}^{-1} B, \tag{17}$$

with $(B_g)_{:,J} = I_r$. The identity block need not be stored explicitly. The A-side construction is symmetric: if I indexes nonsingular pivot rows $A_{I,:}$, choosing $G = A_{I,:}^{-1}$ yields $(A_g)_{I,:} = I_r$ and $B_g = A_{I,:}B$. In practice the pivot structure is fixed by a numerically stable LU-based procedure so unpacking and regauging remain deterministic.

Proposition 1 (intrinsic floating-point storage). The smooth manifold of exact rank-r matrices in $\mathbb{R}^{m\times n}$ has dimension

$$\dim \mathcal{M}_r = r(m+n-r). \tag{18}$$

On any chart where the selected $r \times r$ pivot minor is nonsingular, either block-identity gauge stores exactly this number of floating-point coefficients.

Proof. The redundant factor pair contains $mr+rn$ continuous coordinates. The free action $(A, B) \mapsto (AG, G^{-1}B)$ has group dimension r^2, giving the quotient dimension $mr + rn - r^2 = r(m+n-r)$ (Mishra et al., 2014). In the B-side chart, A_g stores mr coefficients and the nonidentity part of B_g stores $r(n-r)$, totaling $r(m+n-r)$. The A-side chart stores $(m-r)r + rn$, the same total. The pivot indices are discrete metadata rather than continuous floating-point degrees of freedom. □

Thus the block-identity representation does not merely "compress the factors": within a fixed-rank chart it removes exactly the r^2 continuous gauge redundancy. This also explains why an equal-memory comparison need not be rank matched. Ordinary factors consume $r(m+n)$ stored coefficients, while a gauge-fixed chart consumes $r(m+n-r)$.

Tangent space and gauge-null directions. The differential of the factor map $(A, B) \mapsto AB$ is

$$(\Delta A, \Delta B) \mapsto \Delta W = \Delta AB + A\Delta B, \tag{19}$$

so the tangent space of the fixed-rank manifold is

$$T_W\mathcal{M}_r = \{\Delta AB + A\Delta B : \Delta A \in \mathbb{R}^{m\times r}, \Delta B \in \mathbb{R}^{r\times n}\}. \tag{20}$$

The kernel of this differential is exactly

$$(\Delta A, \Delta B) = (A\Omega, -\Omega B), \qquad \Omega \in \mathbb{R}^{r\times r}, \tag{21}$$

which has dimension r^2. Indeed, the finite path $A(t) = Ae^{t\Omega}$, $B(t) = e^{-t\Omega}B$ keeps W unchanged for all t. Conversely, if $\Delta AB + A\Delta B = 0$, full row rank of B implies that ΔA lies in the column space of A, so $\Delta A = A\Omega$ for some Ω; full column rank of A then gives $\Delta B = -\Omega B$. This is the local geometric counterpart of Proposition 1 and agrees with standard fixed-rank manifold descriptions (Vandereycken, 2013; Mishra et al., 2014).

A-only adaptation is restricted to the tangent subspace $\{ZB\}$ and therefore preserves the row subspace defined by B; B-only adaptation uses $\{AZ\}$ and preserves the column subspace defined by A. Updating both factors spans the full fixed-rank tangent space, modulo the gauge-null directions in Eq. (21).

B.2.1 Independent gauge choice and update coordinates

For an MLP, the gauge need not be chosen identically for every projection. In particular, our implementation allows the gate/up projections to use one base gauge while `down_proj` uses the same or the opposite side. We denote these choices by AA, AB, BA, and BB, where the first letter is the gauge side for gate/up and the second is the side for down. In exact arithmetic these are alternative charts for the same rank-constrained operators. In deployed BF16/FP16 arithmetic, however, the implicit identity block, free stored coordinates, and intermediate latent magnitudes differ, so the realized compressed models need not be bitwise or functionally identical.

The gauge also determines which single-factor update has a no-regauge path. Under an A-side gauge, A_g contains an implicit identity pivot block while B_g is stored in full coordinates, so a B-only step leaves the canonical identity block untouched. Under a B-side gauge, the roles reverse and an A-only step has the no-regauge path. For generic factor gradients G_A and G_B, these steps induce

$$\Delta W_A = -\eta G_A B, \qquad \Delta W_B = -\eta A G_B, \tag{22}$$

even before line search. Thus gauge fixing is not merely a storage convention once Euclidean gradients in factor coordinates are used.

The orientation also has a useful SwiGLU interpretation. Gate/up project from the residual stream into the MLP intermediate space, whereas `down_proj` maps back. Consequently, A-only and B-only updates exchange which semantic side of the MLP is held fixed when the operator orientation reverses. The AA/AB/BA/BB experiment therefore probes a structured choice of residual-stream versus intermediate-space update coordinates, rather than four algebraically unrelated implementations.

B.3 A-SIDE VERSUS B-SIDE GAUGE IN PYTORCH-LIKE PSEUDOCODE

The side name refers to *which factor contains the canonical identity block*, not which factor is updated. This is the main convention to keep in mind:

Gauge	Canonical identity	Fully stored factor	No-regauge single-factor step
A-side	$(A_g)_{I,:} = I_r$	B_g	B-only
B-side	$(B_g)_{:,J} = I_r$	A_g	A-only

Thus "B-side gauge + factor A"—our principal configuration—means that the identity block is fixed in B_g, while A_g is stored in full and can be updated without disturbing the identity block. Here I and J are not arbitrary subsets: they are the ordered LU-selected pivot rows/columns that define a nonsingular $r \times r$ chart. Specifically, $I \subset \{0, \dots, m-1\}$ contains r pivot rows of A, whereas $J \subset \{0, \dots, n-1\}$ contains r pivot columns of B. We obtain I by partial-pivoted LU of the tall matrix $A \in \mathbb{R}^{m \times r}$ and obtain J by applying the same row-pivot routine to $B^\top \in \mathbb{R}^{n \times r}$. PyTorch returns LAPACK-style one-based row-swap pivots, so the first listing explicitly converts the swap sequence into the final row permutation. The first r entries of that permutation are the ordered skeleton indices.

Operationally, we set `I = lu_pivot_rows(A)` and `J = lu_pivot_rows(B.T)` using the routine below. Thus I and J are the *pivot index sets* (or skeleton indices), whereas $P_A = A[I, :]$ and $P_B = B[:, J]$ are the corresponding $r \times r$ *pivot blocks*. Complementary indices are stored as metadata where needed. In production, LU factorizations are batched and the small swap-to-permutation conversion follows one host synchronization. For a shared gate/up A-side group, gate- and up-derived candidate pivot sets are both evaluated and we choose the one with smaller worst coefficient growth; for a shared B-side group, the same LU rule is applied to the transpose of the concatenated private B factors. The selected offline pivots are *held fixed* during request-time line search and regauging; we do not repivot per request.

Listing 1: Offline LU pivot selection. PyTorch returns row swaps; converting them to a permutation gives pivot rows I for A, while running the same routine on $B^\top$ gives pivot columns J for B.

```
def lu_pivot_rows(M):
    # M: [N, r], N >= r, full column rank.
    _, swaps, info = torch.linalg.lu_factor_ex(M.float())
    if torch.any(info != 0):
        raise RuntimeError("singular pivot block")

    # LAPACK/cuSOLVER gives 1-based row swaps, not row IDs.
    perm = torch.arange(M.shape[0], device=M.device)
    for k in range(M.shape[1]):
        j = int(swaps[k].item()) - 1
        tmp = perm[k].clone()
        perm[k] = perm[j]
        perm[j] = tmp
    return perm[:M.shape[1]]
```

With I or J fixed, canonicalization is just a solve with the selected pivot block. The following pseudocode is schematic and omits batching and mixed-precision bookkeeping; it uses `torch.linalg.solve` rather than explicitly forming M^{-1}.

Listing 2: Canonical A-side and B-side gauges using the offline pivot sets from Listing 1; factor A has shape (m, r) and factor B has shape (r, n).

```
# W = A @ B; never form dense W during deployment.
def gauge_A(A, B, I):
    # A-side means: pivot rows of A become the identity.
    P = A[I, :]                          # [r, r], nonsingular
    A_g = torch.linalg.solve(P.T, A.T).T          # A @ P^{-1}
    B_g = P @ B                          # P @ B
    # now A_g[I, :] == I_r
    # Store A_g off-pivot rows + all of B_g.
    return A_g, B_g
def gauge_B(A, B, J):
    # B-side means: pivot columns of B become the identity.
    P = B[:, J]                          # [r, r], nonsingular
    A_g = A @ P
    B_g = torch.linalg.solve(P, B)                # P^{-1} @ B
    # now B_g[:, J] == I_r
    # Store all of A_g + B_g off-pivot columns.
    return A_g, B_g
```

Both factors remain valid optimization variables under either gauge. The asymmetry is only computational: if the identity-bearing factor is held fixed, updating the other factor preserves the canonical chart automatically. Updating the identity-bearing factor is equally valid, but it changes the pivot block and is therefore followed by an algebraically equivalent regauging step:

In Listing 3, `gA` and `gB` denote the proposed factor gradients after any balancing, while `eta` denotes the current scalar step; the outer KL-bisection scale is omitted for clarity.

For an A+B step we update both factors and then apply the same restoration associated with the chosen gauge side. Regauging is an exact coordinate change in exact arithmetic; for example, on the B-side branch, $A_g B_{\text{trial}} = (A_g P)(P^{-1} B_{\text{trial}})$. No dense $m \times n$ weight matrix is reconstructed. For MLP cross-gauges, the routine is selected independently per projection group: AA uses A-side for gate/up and down; AB uses A-side for gate/up and B-side for down; BA reverses those choices; and BB uses B-side throughout. Thus BB gives factor A the no-regauge single-factor path in both projection groups, whereas AA gives that path to factor B. Either factor can still be updated in every gauge. This is the algorithmic counterpart of Table 2.

Listing 3: One factor step and restoration of the canonical gauge. Both factors are always updatable; the no-regauge branches explain the runtime reversal in the AA/AB/BA/BB experiments.

```
def step_in_gauge(A_g, B_g, side, factor, gA, gB, eta, I=None, J=None):
    if side == "B" and factor == "A":
        # B_g[:, J] = I is untouched: no regauge needed.
        return A_g - eta * gA, B_g
    if side == "B" and factor == "B":
        B_trial = B_g - eta * gB
        P = B_trial[:, J]                # new pivot block
        A_new = A_g @ P
        B_new = torch.linalg.solve(P, B_trial)        # regauge
        return A_new, B_new
    if side == "A" and factor == "B":
        # A_g[I, :] = I is untouched: no regauge needed.
        return A_g, B_g - eta * gB
    if side == "A" and factor == "A":
        A_trial = A_g - eta * gA
        P = A_trial[I, :]                # new pivot block
        A_new = torch.linalg.solve(P.T, A_trial.T).T  # regauge
        B_new = P @ B_g
        return A_new, B_new
```

Finite-precision gauge sensitivity. The exact gauges in Eq. (15) are mathematically equivalent, but storage is finite precision. Let $A_Q = AQ$, $B_Q = Q^{-1}B$, and suppose stored or materialized factors

are $\widetilde{A}_Q = A_Q + E_A$ and $\widetilde{B}_Q = B_Q + E_B$. Then

$$\widetilde{A}_Q\widetilde{B}_Q - AB = E_AB_Q + A_QE_B + E_AE_B. \tag{23}$$

For any submultiplicative matrix norm,

$$\|\widetilde{A}_Q\widetilde{B}_Q - AB\| \le \|E_A\|\|B_Q\| + \|A_Q\|\|E_B\| + \|E_A\|\|E_B\|. \tag{24}$$

Under the simple normwise rounding model $\|E_A\| \le u\|A_Q\|$ and $\|E_B\| \le u\|B_Q\|$, standard in first-order floating-point analysis (Higham, 2002),

$$\|\widetilde{A}_Q\widetilde{B}_Q - AB\| \le (2u + u^2)\|A_Q\|\|B_Q\| \le (2u + u^2)\kappa(Q)\|A\|\|B\|, \tag{25}$$

where $\kappa(Q) = \|Q\|\|Q^{-1}\|$ in the same induced norm (or the analogous mixed Frobenius/spectral bound). This is only a conservative perturbation bound—the implicit identity block can reduce actual storage error—but it explains why exact gauge equivalence does not imply identical BF16/FP16 realization. It also motivates the LU growth, pivot, factor-rounding, and operator-probe diagnostics used in our implementation.

B.4 Request-local loss-aware adaptation

For a test request x, let $\mathcal{M}(x)$ be the set of causal-LM target positions used for adaptation. Our GSM8K protocol uses demonstration-answer tokens together with observed query tokens, never the unknown answer. The masked causal loss is

$$\mathcal{L}_{\text{req}}(\theta; x) = -\frac{1}{|\mathcal{M}(x)|}\sum_{t\in\mathcal{M}(x)} \log p_\theta(x_t \mid x_{<t}). \tag{26}$$

Each request begins from the same compressed checkpoint. We compute one gradient step in the deployed reduced-rank coordinates, perform constrained line search, generate the answer, and then restore the checkpoint. Thus the reported experiments measure request-local adaptation rather than accumulation across test examples.

B.5 Gauge-balanced factor update and induced operator geometry

For a factorized operator $W = AB$, let $H = \partial\mathcal{L}/\partial W$. The factor gradients are

$$G_A = HB^\top, \qquad G_B = A^\top H. \tag{27}$$

A scalar balance parameter $\gamma > 0$ defines temporary optimization coordinates

$$\bar{A} = \gamma A, \qquad \bar{B} = B/\gamma, \tag{28}$$

which represent the same operator. Ordinary Euclidean SGD in $(\bar{A}, \bar{B})$ maps back exactly to

$$\Delta A = -\eta\frac{G_A}{\gamma^2}, \qquad \Delta B = -\eta\gamma^2 G_B. \tag{29}$$

To see this, note that $G_{\bar{A}} = G_A/\gamma$ and $G_{\bar{B}} = \gamma G_B$ by Eq. (27); one SGD step in barred coordinates followed by the inverse rescaling gives Eq. (29). Thus factor balancing is exactly a scalar optimization-gauge choice, not an ad hoc post-hoc weighting.

The implemented `grad` rule follows from equalizing relative first-order SGD rates. Define

$$r_A = \frac{\|G_A\|_F}{\|A\|_F}, \qquad r_B = \frac{\|G_B\|_F}{\|B\|_F}. \tag{30}$$

In the barred coordinates these become r_A/γ^2 and $\gamma^2 r_B$. Equality therefore requires

$$\boxed{\gamma = \left(\frac{r_A}{r_B}\right)^{1/4}}. \tag{31}$$

For shared-factor groups, the implementation uses the same derivation with aggregate Frobenius norms over the private factors and gradients.

Factor steps are operator-space preconditioners. Combining Eqs. (27)–(29), an A-only step induces the exact operator change

$$\boxed{\Delta W_A = -\frac{\eta}{\gamma^2} H(B^\top B),} \tag{32}$$

and a B-only step induces

$$\boxed{\Delta W_B = -\eta\gamma^2 (AA^\top) H.} \tag{33}$$

Thus A-only and B-only Euclidean factor descent are respectively right- and left-preconditioned operator-gradient steps. If both factors are stepped simultaneously, the exact change is

$$\Delta W = -\eta \left(\frac{1}{\gamma^2} H B^\top B + \gamma^2 A A^\top H \right) + \eta^2 H B^\top A^\top H. \tag{34}$$

The final term is second order in the step size. Under a general gauge $(A, B) \mapsto (AQ, Q^{-1}B)$, the two preconditioners become $AQQ^\top A^\top$ and $B^\top Q^{-\top} Q^{-1} B$. Hence Euclidean factor descent is invariant to orthogonal gauge changes but not to arbitrary $Q \in \mathrm{GL}(r)$. This gives a direct theoretical reason for the gauge/update-coordinate ablations in Appendices C.1 and C.2.

Our principal GSM8K configuration uses a B-side gauge and factor-A adaptation, for which the stored full factor A is updated while B remains fixed. This admits a memory-efficient path that never materializes all full factors for all layers simultaneously.

B.6 Gauge-invariant operator projection through a small core

For reference, we also implemented a proposal defined in operator space rather than in factor coordinates. If $X \in \mathbb{R}^{t\times n}$ is the captured layer input and $D \in \mathbb{R}^{t\times m}$ is the gradient with respect to the layer output, then $H = D^\top X$ and the unconstrained operator step is

$$W_{\mathrm{trial}} = AB - \eta D^\top X. \tag{35}$$

It has the exact skinny factorization

$$W_{\mathrm{trial}} = \begin{bmatrix} A & -\sqrt{\eta} D^\top \end{bmatrix} \begin{bmatrix} B \\ \sqrt{\eta} X \end{bmatrix} =: LR. \tag{36}$$

Take thin QR factorizations $L = Q_L R_L$ and $R^\top = Q_R R_R$, giving

$$W_{\mathrm{trial}} = Q_L K Q_R^\top, \qquad K = R_L R_R^\top. \tag{37}$$

Because Q_L and Q_R have orthonormal columns, K and W_{trial} have the same nonzero singular values, with singular vectors mapped by Q_L and Q_R. If $K = U\Sigma V^\top$, the Eckart–Young theorem (Eckart & Young, 1936) therefore gives the exact best rank-r approximation

$$\boxed{\Pi_r(W_{\mathrm{trial}}) = Q_L U_r \Sigma_r V_r^\top Q_R^\top.} \tag{38}$$

This requires an SVD only of the small core K, a standard QR/SVD reduction in numerical linear algebra (Golub & Van Loan, 2013). The construction is mathematically equivalent, up to floating-point roundoff, to materializing W_{trial} and truncating its SVD; metric projection/truncated SVD is also a standard retraction mechanism for fixed-rank manifold optimization (Vandereycken, 2013). Since the proposal depends only on $W = AB$ and H, not on a particular (A, B) chart, it is gauge invariant before the result is returned to the deployment gauge.

This operator-projected construction is useful as a mathematical control for the A/B-coordinate experiments, although its QR/SVD cost is substantially higher than the fully stored-factor fast path. Appendix C.2 reports the preliminary ablations and the provenance limitation of the early operator-projected task result.

B.7 Hard functional KL trust region

A low request loss is only a surrogate for downstream generation quality. We therefore constrain the functional change from the pre-update compressed model. Let p_0 be its next-token distribution on

a fixed KL token set and p_λ the distribution after scaling a proposed direction d by $\lambda \in [0, 1]$. We require

$$D_{\mathrm{KL}}(p_0 \| p_\lambda) \le \varepsilon. \tag{39}$$

Locally, KL divergence induces the Fisher information metric (Amari, 1998; Schulman et al., 2015). Under the usual smoothness conditions,

$$D_{\mathrm{KL}}(p_\theta \| p_{\theta+\lambda d}) = \frac{\lambda^2}{2} d^\top F d + O(\lambda^3), \tag{40}$$

where F is the predictive Fisher on the same evaluation contexts. Thus the quadratic approximation would select

$$\lambda_{\mathrm{quad}} \approx \sqrt{\frac{2\rho\varepsilon}{d^\top F d}} \tag{41}$$

to reach target fraction $\rho\varepsilon$. Our deterministic KL bisection instead evaluates the true nonlinear model and searches for the largest admissible scale near that target. In the local regime Eq. (40) explains why scaling a fixed direction is approximately monotone in λ^2; outside that regime, direct KL evaluation avoids relying on the quadratic approximation.

This procedure separates *where* the model adapts from the total functional change, allowing block-selection policies to be compared at nearly matched realized KL.

B.8 Request-adaptive block selection

Partition the adaptable coordinates into transformer blocks $b = 1, \dots, B$. Let g_b denote the request gradient in the temporary optimization coordinates after factor balancing. The implementation aggregates its squared Frobenius norm over adaptable projections in each block,

$$s_b = \|g_b\|_2 = \left(\sum_{\ell \in b} \|G_\ell^{(\mathrm{opt})}\|_F^2 \right)^{1/2}. \tag{42}$$

Hard top-k selection activates only the k highest-scoring blocks,

$$w_b = \mathbf{1}[b \in \mathrm{TopK}(s_1, \dots, s_B; k)]. \tag{43}$$

A soft alternative uses

$$w_b = \left(\frac{s_b}{\max_j s_j} \right)^p, \tag{44}$$

with exponent $p > 0$. The final proposed update is blockwise scaled by w_b and then globally scaled by the KL-bisection coefficient λ. Hard selection is especially attractive in the current implementation because rejected blocks can discard their update state before KL search, reducing both memory and adaptation time.

Proposition 2 (local KL-constrained support selection). Let z stack the adaptable optimization coordinates, let $g = \nabla_z \mathcal{L}$, and approximate the local KL by $\frac{1}{2}\delta^\top F \delta$ with a positive-definite (or damped) Fisher matrix F. For a fixed coordinate support S, the first-order problem

$$\min_{\delta_S} \; g_S^\top \delta_S \quad \text{s.t.} \quad \frac{1}{2} \delta_S^\top F_S \delta_S \le \varepsilon \tag{45}$$

has solution

$$\boxed{\delta_S^\star = -\sqrt{\frac{2\varepsilon}{g_S^\top F_S^{-1} g_S}}\, F_S^{-1} g_S} \tag{46}$$

and maximum first-order loss decrease

$$\boxed{-g_S^\top \delta_S^\star = \sqrt{2\varepsilon\, g_S^\top F_S^{-1} g_S}.} \tag{47}$$

Proof. Let $y = F_S^{1/2} \delta_S$ and $a = F_S^{-1/2} g_S$. The constraint is $\|y\|_2 \le \sqrt{2\varepsilon}$ and the linear objective is $a^\top y$. Cauchy–Schwarz is tight for $y = -\sqrt{2\varepsilon}\, a / \|a\|_2$, which maps back to Eq. (46); substituting gives Eq. (47). □

Under a block-diagonal, common-curvature approximation $F_b \approx cI$ with the same $c > 0$ across blocks,

$$g_S^\top F_S^{-1} g_S \approx \frac{1}{c} \sum_{b \in S} \|g_b\|_2^2. \tag{48}$$

Therefore, under a cardinality constraint $|S| = k$, selecting the k largest gradient norms is exactly optimal for this local surrogate. If curvature scales differ, the corresponding idealized block utility is $g_b^\top F_b^{-1} g_b$ (or $\|g_b\|^2/c_b$ in an isotropic per-block model), which clarifies both the approximation behind hard top-k gradient selection and the motivation for more expensive curvature/functional scores. Our method deliberately uses the inexpensive gradient approximation and relies on the global hard KL search to control the realized functional displacement.

B.9 Why the trust region does not solve support selection by itself

Suppose two update policies produce directions d_1 and d_2 that are each scaled to satisfy the same local KL radius. By Eq. (40), matching the scalar quadratic form $\frac{1}{2}\delta^\top F \delta$ controls the magnitude of functional displacement but not its direction. Therefore two policies with nearly identical realized KL can have different downstream effects. Proposition 2 further shows that, even in the local quadratic model, the attainable first-order surrogate gain depends on the chosen support through $g_S^\top F_S^{-1} g_S$. This is exactly the regime probed by all-block versus selective adaptation.

B.10 A local adaptation-access ratio

The scale/compression experiments suggest an intuitive "recoverability" regime, but exact-match reasoning is nonlinear and cannot be inferred from a one-step local theorem. We therefore introduce only a local diagnostic quantity. Let $\mathcal{S}$ be an allowable deployment-update subspace and let $\mathcal{U} \supseteq \mathcal{S}$ be a richer reference space at the same model state. Under the same positive-definite/damped Fisher approximation, define

$$q(\mathcal{S}) = g_{\mathcal{S}}^\top F_{\mathcal{S}}^{-1} g_{\mathcal{S}}, \qquad \mathcal{R}_{\mathrm{local}}(\mathcal{S};\mathcal{U}) = \frac{q(\mathcal{S})}{q(\mathcal{U})}. \tag{49}$$

Because Eq. (47) is the optimum over each nested subspace, $0 \le \mathcal{R}_{\mathrm{local}} \le 1$, and the ratio of their maximum local first-order loss reductions is $\sqrt{\mathcal{R}_{\mathrm{local}}}$. This quantity measures how much of the locally useful request direction remains accessible to the deployed update coordinates. It is *not* claimed to predict GSM8K exact-match accuracy, nor is it estimated in our current experiments. Severe compression can also change the model state itself enough that a local request-loss direction is no longer aligned with correct reasoning, consistent with the observed failure of prompt-loss improvement to guarantee task recovery.

C Detailed Experimental Evidence

C.1 Matched deterministic gauge and update-coordinate matrix

The final matched mechanism study uses the same principal Qwen3-0.6B deployment protocol as the decomposition comparison: 4,096-token WikiText calibration, stored-parameter fraction 0.95, full 1,319-example GSM8K evaluation, all-block adaptation, gradient balancing, and hard KL budget 0.025 with deterministic bisection. This study supersedes the earlier C4-calibrated matrix for accuracy comparisons.

The realized KL varies only from 0.02035 to 0.02284, while downstream changes range from -2.81 to $+2.65$ points. This is a direct controlled example in which scalar functional displacement is insufficient to determine downstream behavior. It also exposes the predicted storage-coordinate asymmetry: under AA, B-only updates preserve the canonical gauge without regauging, whereas under BB the analogous no-regauge path is A-only. BB is the only gauge where A, B, and A+B all improve task accuracy. Although BB+A+B has the highest all-block exact match, BB+A attains 34.27% at roughly one fifth of the adaptation time, motivating the practical choice used elsewhere in the paper.

Prompt fit is also insufficient: BA+A improves CE by 0.1919 but loses 0.23 points, whereas BB+A+B improves CE less than BB+A yet has slightly higher exact match.

Table 5: Matched gauge × update-coordinate matrix. ΔAcc is relative to the fixed model under the same gauge; ΔCE is masked prompt cross-entropy improvement. All adaptive rows use the same nominal hard KL budget 0.025.

Gauge	Update	Fixed acc. (%)	Adapted acc. (%)	ΔAcc (pp)	ΔCE	KL
AA	A	32.22	32.75	+0.53	0.1947	0.02217
AA	B	32.22	30.40	-1.82	0.0205	0.02035
AA	A+B	32.22	29.87	-2.35	0.0570	0.02087
AB	A	32.68	33.66	+0.99	0.1979	0.02207
AB	B	32.68	32.37	-0.30	0.1322	0.02195
AB	A+B	32.68	33.97	+1.29	0.1592	0.02180
BA	A	31.84	31.61	-0.23	0.1919	0.02284
BA	B	31.84	29.64	-2.20	0.0211	0.02103
BA	A+B	31.84	29.04	-2.81	0.0526	0.02128
BB	A	32.07	34.27	+2.20	0.2348	0.02274
BB	B	32.07	33.97	+1.90	0.1436	0.02231
BB	A+B	32.07	**34.72**	**+2.65**	0.1537	0.02220

Table 6: Implementation cost for the same matched matrix. Times are measured request-local TTA seconds per GSM8K problem on the recorded A40 runs; they are not optimized serving throughput. Acceptance is the fraction of request-local TTA steps accepted by the line search.

Gauge	Update	TTA s/problem	Acceptance
AA	A	6.481	1.000
AA	B	**1.208**	0.988
AA	A+B	6.602	1.000
AB	A	5.044	1.000
AB	B	3.233	1.000
AB	A+B	7.020	1.000
BA	A	2.857	1.000
BA	B	4.370	0.998
BA	A+B	6.013	1.000
BB	A	**1.332**	1.000
BB	B	6.266	1.000
BB	A+B	6.428	1.000

C.2 Preliminary gauge and update-coordinate ablations

Before freezing the final B/B + factor-A configuration, we ran a sequence of gauge-side and update-coordinate ablations. We preserve them because they expose a useful property of the method: two factorizations can represent the same rank-r operator in exact arithmetic while inducing different finite-precision deployment error, gradient coordinates, and online-update cost. These historical mechanism experiments predate the matched matrix in Appendix C.1; they show how the final experiment was motivated, while the matched WikiText-calibrated matrix is the principal gauge/update accuracy comparison.

Dual gauge side changes both optimization geometry and cost. Table 7 summarizes a corrected 64-block WikiText experiment on Qwen3-0.6B at stored-parameter fraction 0.95 with 4,096-token C4 calibration, 128 prompt tokens, 256 decode tokens, tail-half adaptation, unpack/regauge, gradient balancing, and hard KL budget 0.05. Under the A-side gauge, B is the factor stored in full coordinates; under the B-side gauge, A is stored in full coordinates. The no-regauge update therefore swaps sides with the storage gauge.

The striking comparison is not simply A versus B as abstract factors. The best row, B/B + A, uses an A-only update that preserves the B-side gauge without regauging, while also yielding the largest perplexity improvement in this matrix. Conversely, A/A + B has the analogous no-regauge property

Table 7: Preliminary WikiText dual-gauge/factor ablation. Lower NLL is better. "A+B" updates both factors before returning to the fixed gauge. Times are historical per-block TTA measurements and are included to show the no-regauge runtime asymmetry, not optimized serving throughput.

Gauge	Updated factor	Fixed NLL	Adapted NLL	ΔPPL (%)	TTA s/block
A/A	A	3.2027	3.1525	-4.90	≈1.95
A/A	B	3.2027	3.1933	-0.94	≈0.34
A/A	A+B	3.2027	3.1833	-1.92	≈2.40
B/B	A	3.2044	**3.1510**	**-5.20**	≈0.32
B/B	B	3.2044	≈3.166	-3.75	≈1.95
B/B	A+B	3.2044	≈3.164	-3.97	≈2.05

but produces a much weaker improvement. Both factors are valid update coordinates in either gauge; these results show that computational convenience and task-gradient geometry are distinct, and the two no-regauge directions are not functionally equivalent.

The down-projection gauge can be chosen independently. We next allowed gate/up and down projections to use different gauge sides. With factor-A adaptation and otherwise matched WikiText settings, the four AA/AB/BA/BB choices gave the adapted NLLs in Table 8. The same-side B/B configuration was best, but the B/A cross gauge was nearly tied with A/A, showing that an "opposite" down gauge is a legitimate design degree of freedom rather than an algebraic inconsistency.

Table 8: Preliminary independent gate/up–down gauge experiment on WikiText. All rows use factor-A unpack/regauge at hard KL budget 0.07; lower NLL is better.

Gauge (gate/up, down)	A/A	A/B	B/A	B/B
Adapted NLL	3.14353	3.14605	3.14349	**3.14056**

A corrected full-test GSM8K matrix later showed a larger fixed-model effect from gauge realization (Table 9). For each gauge we report the fixed accuracy and the best TTA row among factor $\{A, B, A{+}B\}$ and hard KL $\{0.03, 0.04, 0.05\}$. This table is descriptive of the preliminary sweep; it is not used to tune the final main-paper setting.

Table 9: Corrected preliminary full-test GSM8K AA/AB/BA/BB matrix (Qwen3-0.6B, 1,319 examples, stored-parameter fraction 0.95). "Best TTA" is the best row inside the historical factor/KL matrix for that gauge.

Gauge	Fixed acc. (%)	Best TTA acc. (%)	Factor	hard KL budget
A/A	31.54	32.60	A	0.03
A/B	31.54	33.13	B	0.05
B/A	34.12	33.74	A	0.04
B/B	**34.72**	34.27	A	0.05

The fixed spread from 31.54% to 34.72% is important: although the gauges are equivalent algebraically before finite-precision storage, their deployed BF16 realizations are not identical. The same experiment also warned against selecting update strength by prompt fit. In a subsequent B/B + factor-A hard KL diagnostic sweep, exact-match accuracy was 34.87%, 35.03%, **36.39%**, and 34.27% at KL budgets 0.005, 0.01, 0.02, and 0.03, respectively. The prompt-CE improvement increased from 0.1146 at KL 0.02 to 0.1789 at KL 0.03 even as accuracy dropped by 2.12 points. This observation directly motivated the later trust-region and support-selection analysis.

Factor-coordinate updates versus operator-space projection. Factor-A and factor-B steps are intentionally coordinate dependent, as Eq. (22) makes explicit. We also implemented the gauge-independent operator-projected control in Eq. (36). The early full-test GSM8K operator-projected run is useful only as a *preliminary diagnostic*: because shell-local sweep variables did not propagate

through Slurm, the nominal AA/AB/BA/BB directory labels were invalid and all jobs effectively used the same A/A, rank-mode=rank, rank-0.90, both-factor configuration with a soft KL penalty ($\beta = 3$). Across those nominally separate repetitions, fixed accuracy lay between 25.63% and 28.13%, while operator-projected TTA reached 29.49–30.63%, with realized KL about 0.067–0.070 and nearly 100% acceptance. We therefore retain the run only as evidence that the auxiliary operator-space construction can change task predictions under that setup, **not** as evidence for one gauge over another or as a matched comparison against the final hard KL factor update.

This operator-space reference is nevertheless mathematically useful. It separates two questions that are otherwise conflated: (i) which operator direction is favored by the request loss, and (ii) which factor coordinates provide an efficient constrained realization of that direction. The QR/small-core-SVD projection makes the first question gauge independent, whereas A/B unpack-regauge exposes the second. A later 128-example runtime gate, retained only as a C4-calibrated mechanism diagnostic, measured 125.18–125.41 s/problem for operator projection versus 1.373 s/problem for B/B + factor-A—about 91× slower. Operator-projected accuracy was 20.31% under AA and 17.97% under BB, versus 20.31% for the practical BB factor-A control, providing no indication of a task-level advantage in that small diagnostic. Because the calibration corpus differs from the principal paper protocol, these accuracies are not used for main-paper comparisons; the runtime gate motivated our decision not to run a full operator-projected evaluation within the computational budget.

C.3 Decomposition method comparison at a fixed stored-memory budget

Table 10 records the completed deterministic GSM8K comparison at stored-parameter fraction 0.95. SVD, ASVD, and SVD-LLM use their native ungauged factors; LatentLLM (Koike-Akino et al., 2026) uses the prior gauge-fixed deployment representation. The comparison is intentionally memory-budget matched rather than rank matched because deployment memory is the controlled resource.

Table 10: GSM8K exact-match accuracy (%) at stored-parameter fraction 0.95. TTA uses one request-local step and hard KL budget 0.025.

Model	Method	Fixed	All-block	Top-50% blocks
Qwen3-0.6B	SVD	0.00	0.00	0.00
	ASVD	0.83	1.21	1.21
	SVD-LLM	1.67	3.03	2.88
	LatentLLM	32.07	34.27	**34.65**
Qwen3-1.7B	SVD	0.00	0.00	0.00
	ASVD	19.86	21.46	21.83
	SVD-LLM	14.71	15.54	15.09
	LatentLLM	61.94	63.53	**64.59**

For LatentLLM, hard selection improves over the same compressed baseline by $+2.58$ percentage points on Qwen3-0.6B and $+2.65$ points on Qwen3-1.7B. Exact paired McNemar tests give $p = 0.02934$ and $p = 0.007551$, respectively. ASVD also benefits from selective TTA at 1.7B ($+1.97$ points, $p = 0.02437$), indicating that request-adaptive support is not exclusive to the gauge-fixed representation. Nevertheless, LatentLLM retains substantially higher absolute task accuracy under the same stored-memory budget.

C.4 Frozen cross-task transfer

Table 11 records the final frozen transfer experiments that are summarized in the main text. All compressed rows use the principal B/B + factor-A representation at stored-parameter fraction 0.95, one request-local step, gradient balancing, hard KL budget 0.025, and deterministic KL bisection; no task-specific TTA tuning is introduced.

For WikiText, all-block TTA gives the best held-out likelihood, while top-k retains 91–92% of the NLL reduction at lower adaptation cost. The ARC-chat evaluator follows the canonical zero-shot generation protocol and was parity-checked against an independent Language Model Evaluation Harness run:

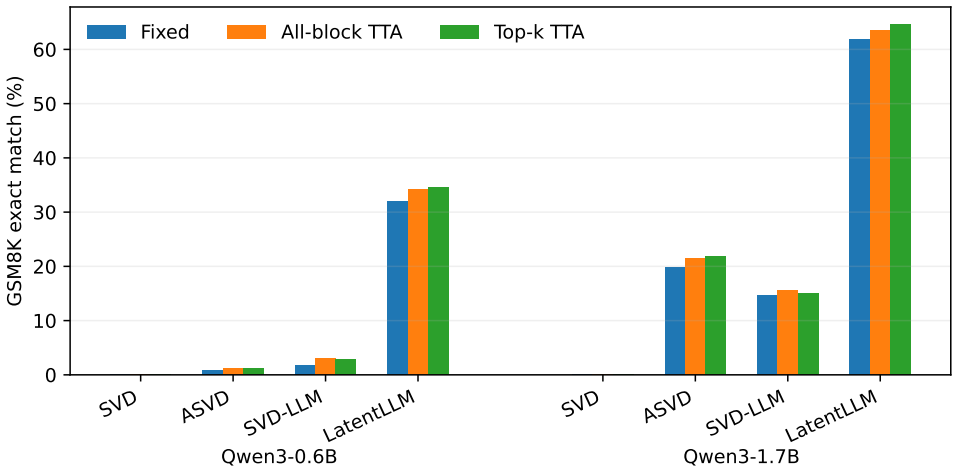

Figure 3: Decomposition-method comparison at a common stored-parameter fraction of 0.95. Values match the frozen experiments summarized in Table 10.

Table 11: Frozen WikiText and ARC-Challenge chat transfer. WikiText reports NLL; ARC reports exact-match generation accuracy.

Task/model	Policy	Metric	KL	ΔCE	Blocks	TTA s
WikiText 0.6B	Fixed	3.17467 NLL	–	–	–	0.000
	All	3.11950 NLL	0.02266	0.4348	28/28	1.335
	Top-k	3.12427 NLL	0.02243	0.4068	14/28	1.096
WikiText 1.7B	Fixed	2.94770 NLL	–	–	–	0.000
	All	2.83807 NLL	0.02236	0.4197	28/28	2.766
	Top-k	2.84690 NLL	0.02218	0.4059	14/28	1.981
ARC-chat 1.7B	Dense	73.12% EM	–	–	–	0.000
	Fixed	66.21% EM	–	–	–	0.000
	All	65.78% EM	0.01602	0.3330	28/28	2.643
	Top-k	66.13% EM	0.01597	0.3412	14/28	1.957

dense accuracy is 73.12% here versus 72.53% $\pm$ 1.30 points independently. Compression reduces the model to 66.21%, leaving meaningful headroom, but the frozen request-local update does not recover it despite substantial prompt-CE improvement.

Paired ARC-chat transitions make the neutral result explicit. Fixed versus all-block gives 17 wrong→right and 22 right→wrong changes (exact McNemar $p = 0.522397$); fixed versus top-k gives 19 and 20 ($p = 1.0$); and all-block versus top-k gives 21 and 17 ($p = 0.627103$). Thus top-k changes 39/1,172 predictions relative to fixed with almost perfectly balanced beneficial and harmful flips. This is evidence of surrogate–task mismatch, not evidence that the optimization failed to reduce its intended request-local objective.

C.5 C4-CALIBRATED WIKITEXT SCALE SWEEP

To separate likelihood recovery from task-specific exact-match effects, we calibrate the offline low-rank subspace from the first 4,096 C4 tokenizer tokens and evaluate request-local adaptation on 64 deterministic WikiText-2-raw-v1 test continuation blocks. All compressed cells use B/B gauge, factor-A unpack/regauge, one step at learning rate 0.0015, gradient balancing, hard KL budget 0.025 with deterministic bisection, and the same 128-token prompt / 256-token continuation structure. No fraction- or model-specific retuning is used. The dense row uses the identical continuation scorer without decomposition or TTA.

Table 12: C4-calibrated → WikiText perplexity across stored-parameter fraction. Lower is better. "Dense" denotes the static undecomposed model under the same continuation scorer.

Model / fraction	Dense	Fixed	All	Top-k
Qwen3-1.7B / 0.70	17.730	58.292	50.343	50.866
Qwen3-1.7B / 0.75	17.730	44.904	38.658	38.871
Qwen3-1.7B / 0.80	17.730	32.777	29.252	29.431
Qwen3-1.7B / 0.85	17.730	26.419	24.008	24.222
Qwen3-1.7B / 0.90	17.730	22.797	20.468	20.670
Qwen3-1.7B / 0.95	17.730	19.721	**17.677**	17.818
Qwen3-4B / 0.70	14.963	48.908	40.522	40.780
Qwen3-4B / 0.75	14.963	34.436	30.421	30.517
Qwen3-4B / 0.80	14.963	25.933	24.118	24.215
Qwen3-4B / 0.85	14.963	21.046	19.208	19.298
Qwen3-4B / 0.90	14.963	17.700	15.996	16.065
Qwen3-4B / 0.95	14.963	15.711	**14.183**	14.325

Table 13: Adaptation diagnostics for the C4→WikiText scale sweep. Each top-k row updates half the transformer blocks.

Model / frac.	$\mathrm{KL}_{\mathrm{all}}$	$\mathrm{KL}_{\mathrm{topk}}$	$\Delta\mathrm{CE}_{\mathrm{all}}$	$\Delta\mathrm{CE}_{\mathrm{topk}}$	t_{all} (s)	t_{topk} (s)
1.7B / 0.70	.02267	.02257	.4669	.4521	1.994	1.509
1.7B / 0.75	.02236	.02257	.4584	.4563	2.122	1.576
1.7B / 0.80	.02246	.02243	.4060	.3940	2.284	1.661
1.7B / 0.85	.02210	.02270	.4457	.4368	2.456	1.739
1.7B / 0.90	.02228	.02199	.4401	.4268	2.583	1.876
1.7B / 0.95	.02208	.02202	.4243	.4160	2.777	1.931
4B / 0.70	.02247	.02270	.4729	.4588	4.154	2.814
4B / 0.75	.02237	.02235	.3842	.3680	4.455	3.034
4B / 0.80	.02217	.02183	.3568	.3398	4.847	3.242
4B / 0.85	.02214	.02163	.4252	.3995	5.265	3.416
4B / 0.90	.02234	.02214	.4135	.3921	5.609	3.631
4B / 0.95	.02175	.02182	.3742	.3492	5.951	3.846

All-block TTA lowers perplexity at every one of the 12 compressed model/fraction points, by 7.0–17.1% relative to fixed; top-k likewise improves all 12 points by 6.6–16.6%. The top-k curve closely follows all-block despite using half the blocks. At stored-parameter fraction 0.95, all-block reaches 17.677 PPL for Qwen3-1.7B and 14.183 for Qwen3-4B, slightly below the corresponding static dense references of 17.730 and 14.963. Because the compressed model receives request-specific adaptation while the dense reference remains static, this comparison measures request-time compensation and should not be read as a claim of globally superior compressed checkpoints.

C.6 Compression-severity and model-scale sweep

Table 14 uses one frozen TTA configuration across Qwen3-1.7B and Qwen3-4B at six stored-parameter fractions from 0.70 through 0.95. The method is not retuned per model or compression point; "Soft" denotes the soft-gradient allocation of Eq. (44) with $p = 2$.

The denser sweep reveals no single compression threshold at which TTA becomes beneficial. Below 0.80–0.85, the fixed task accuracy itself falls sharply and one local step produces only small absolute changes relative to the lost capability. At the high-accuracy end, especially Qwen3-4B/0.95, little headroom remains. The clearest positive transfer is Qwen3-1.7B/0.95, where top-k gains 2.73 points without per-model retuning.

Only Qwen3-1.7B/0.95 shows a significant positive top-k change at the 0.05 level. A complementary warning comes from all-block TTA at 1.7B/0.80: it changes 72 wrong answers to correct but 104

Table 14: GSM8K exact-match accuracy (%) across model scale and stored-parameter fraction. "Top-k" selects half of transformer blocks by request-gradient score.

Model / fraction	Fixed	All	Top-k	Soft
Qwen3-1.7B / 0.70	0.99	0.99	0.68	0.76
Qwen3-1.7B / 0.75	5.38	5.53	5.53	5.69
Qwen3-1.7B / 0.80	25.25	22.82	23.65	24.18
Qwen3-1.7B / 0.85	41.32	41.55	42.30	41.70
Qwen3-1.7B / 0.90	58.30	58.07	59.74	58.38
Qwen3-1.7B / 0.95	61.94	63.38	64.67	64.59
Qwen3-4B / 0.70	0.91	1.06	1.52	1.36
Qwen3-4B / 0.75	7.73	8.57	9.02	8.64
Qwen3-4B / 0.80	34.12	33.89	33.28	32.90
Qwen3-4B / 0.85	59.14	58.76	59.14	58.23
Qwen3-4B / 0.90	75.89	76.04	76.65	75.36
Qwen3-4B / 0.95	85.14	85.22	85.44	85.06

Table 15: Exact paired comparison of fixed versus top-k for the scale sweep. W→C and C→W denote wrong-to-correct and correct-to-wrong transitions.

Model / fraction	Δacc (pp)	W→C	C→W	p
Qwen3-1.7B / 0.70	-0.30	3	7	0.3437
Qwen3-1.7B / 0.75	+0.15	46	44	0.9161
Qwen3-1.7B / 0.80	-1.59	77	98	0.1303
Qwen3-1.7B / 0.85	+0.99	96	83	0.3698
Qwen3-1.7B / 0.90	+1.44	99	80	0.1783
Qwen3-1.7B / 0.95	+2.73	101	65	0.0064
Qwen3-4B / 0.70	+0.61	11	3	0.0574
Qwen3-4B / 0.75	+1.29	51	34	0.0821
Qwen3-4B / 0.80	-0.83	93	104	0.4763
Qwen3-4B / 0.85	+0.00	103	103	1.0000
Qwen3-4B / 0.90	+0.76	71	61	0.4335
Qwen3-4B / 0.95	+0.30	36	32	0.7163

correct answers to wrong, for -2.43 points and exact McNemar $p = 0.0192$. Top-k reduces the magnitude of that degradation and is not significant. Across all adaptive cells, however, masked prompt CE improves, so request-surrogate reduction alone does not predict task recovery.

Interpretation. The cross-task and scale tables above provide the detailed evidence for two conclusions emphasized in the paper body: request-surrogate improvement is not a sufficient statistic for downstream reasoning, and hard support selection can reduce adaptation cost while changing the functional direction at nearly matched KL. The extended grid further shows that recoverability is a window jointly controlled by retained task structure, remaining headroom, and update direction rather than a single monotone compression threshold.

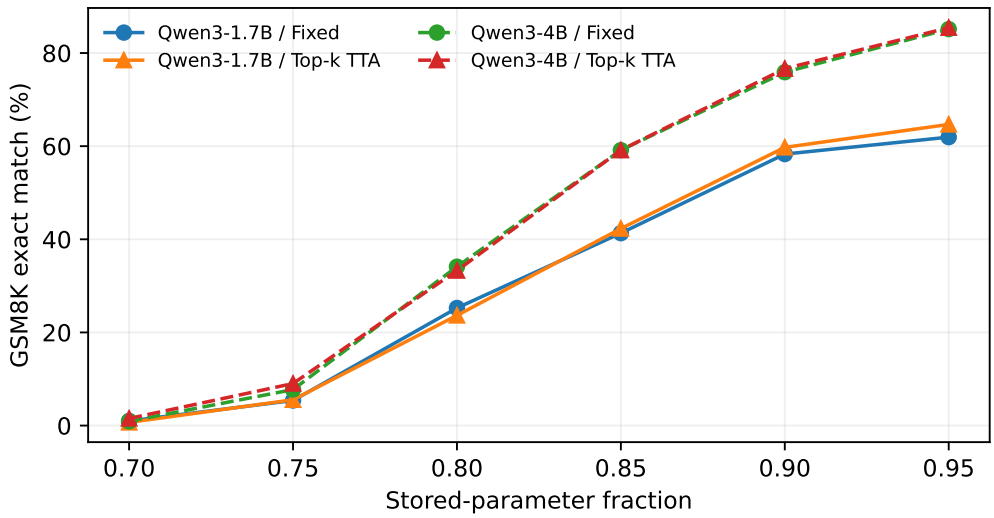

Figure 4: Frozen GSM8K top-k TTA versus the fixed compressed model across Qwen3 scale and stored-parameter fraction. The same TTA hyperparameters are transferred across all points. The six-point 0.70–0.95 grid is retained as a recoverability-boundary diagnostic: it shows that exact-match recovery is model- and budget-dependent rather than uniformly positive.

D REFERENCE GUIDE: ACRONYMS, DATASETS, AND MODELS

This section collects frequently used abbreviations and the external datasets, model checkpoints, and evaluation software used in the experiments. License labels below are informational summaries of the upstream repository metadata checked on September 24, 2026; the upstream license files and source-data terms remain authoritative.

D.1 ACRONYMS AND SHORTHAND

Table 16: Frequently used acronyms, abbreviations, and local shorthand.

Term	Meaning in this paper
A / B / A+B	Update factor A, factor B, or both low-rank factors.
AA / AB / BA / BB	Cross-gauge shorthand: the first letter is the gate/up gauge and the second is the down-projection gauge; each letter is A-side or B-side.
ARC	AI2 Reasoning Challenge.
ASVD	Activation-aware Singular Value Decomposition.
BF16	bfloat16 floating-point format.
C4	Colossal Clean Crawled Corpus.
CE	Cross-entropy.
CUDA	Compute Unified Device Architecture; NVIDIA's GPU programming/runtime platform.
EM	Exact-match accuracy.
FP16	16-bit floating-point format.
$\mathrm{GL}(r)$	General linear group of invertible $r \times r$ matrices.
GSM8K	Grade School Math 8K benchmark.
KL	Kullback–Leibler divergence.
LM / LLM	Language model / large language model.
LoRA	Low-Rank Adaptation.
LU	Lower–upper triangular matrix factorization.
MLP	Multilayer perceptron; here, primarily the transformer feed-forward block.
NLL	Negative log-likelihood.
PEFT	Parameter-efficient fine-tuning.
PPL	Perplexity.
QR	Orthogonal–triangular matrix factorization.
SGD	Stochastic gradient descent.
SVD	Singular value decomposition.
SVD-LLM	Truncation-aware SVD method for LLM compression.
TTA	Test-time adaptation.
TTT	Test-time training.
W8A8	8-bit weights and 8-bit activations.

For the classical online-subspace methods cited in Related Work, PAST denotes Projection Approximation Subspace Tracking, GROUSE denotes Grassmannian Rank-One Update Subspace Estimation, and PETRELS denotes Parallel Subspace Estimation and Tracking by Recursive Least Squares from Partial Observations.

D.2 DATASETS AND EVALUATION TASKS

Table 17: Datasets/evaluation tasks used by the reported experiments.

Resource	Use in this work	License / source note
WikiText-2 raw (Merity et al., 2016)	`Salesforce/wikitext`, configuration `wikitext-2-raw-v1`. The principal GSM8K/decomposition studies use the first 4,096 tokenizer tokens from the training split for offline calibration; the language-model transfer and C4-calibrated scale studies evaluate held-out test continuations.	Upstream metadata lists CC BY-SA 3.0 and GFDL; the current card prose also references CC BY-SA 4.0. See the upstream dataset card and underlying Wikipedia terms.
GSM8K (Cobbe et al., 2021)	Main mathematical-reasoning benchmark; deterministic exact match on the full 1,319-example test split. Training examples are used only as observed demonstrations in the prompting protocol, never as held-out test labels.	MIT License; see the upstream dataset card.
ARC-Challenge (Clark et al., 2018)	Second reasoning benchmark; full 1,172-example test split under the canonical zero-shot chat-generation protocol, with dense parity checked against the Language Model Evaluation Harness.	CC BY-SA 4.0 in the upstream repository metadata; see the upstream dataset card.
C4 (Raffel et al., 2020)	English C4 supplies the 4,096-token offline calibration stream for the main C4→WikiText likelihood-scale study and is also used in historical/auxiliary calibration and the operator-projection runtime diagnostic. It is not the principal calibration corpus for the GSM8K accuracy comparisons.	ODC-BY; the upstream card also states that Common Crawl terms apply to the contained source material. See allenai/c4.

D.3 MODELS AND EVALUATION SOFTWARE

All main experiments use public dense Qwen3 checkpoints (Qwen Team, 2025), loaded in BF16 for offline decomposition. The original dense checkpoint is used to construct the deployed reduced-rank model and is not required by the request-local TTA procedure after deployment.

Table 18: Pretrained model checkpoints used in the experiments.

Checkpoint	Role in this work	License
`Qwen/Qwen3-0.6B`	Decomposition-method comparison, gauge/update-coordinate analysis, and WikiText transfer.	Apache License 2.0 (model card).
`Qwen/Qwen3-1.7B`	Principal scale/compression studies, WikiText transfer, GSM8K, and canonical ARC-Challenge chat transfer.	Apache License 2.0 (model card).
`Qwen/Qwen3-4B`	Larger-scale GSM8K compression study and C4→WikiText likelihood-scale study.	Apache License 2.0 (model card).

For the canonical ARC-Challenge chat protocol, we use and parity-check against the Language Model Evaluation Harness (Biderman et al., 2024); the upstream software is distributed under the MIT License (license file).